ENGR-UH 3511

Computer Organization and Architecture

Name: Nishant Aswani

Net ID: nsa325

Assignment Title: Homework 4

Question 4.12.1

The table shows the number of stall cycles given a RAW hazard type.

RAW hazard	stall cycles
Ex to 1st	2
Mem to 1st	2
Ex to 2nd	1
Mem to 2nd	1
Ex to 1st and Mem to 2nd	2

$$\text{New CPI} = 1 + 2 * (0.05 + 0.2 + 0.1) + 1 * (0.05 + 0.1) = 1.85$$

$$\textbf{Fraction of Stalled Cycles} = \frac{0.85}{1.85} * 100 = 46\%$$

Question 4.12.2

In the case of full forwarding, we can use the following table:

RAW hazard	stall cycles
Ex to 1st	0
Mem to 1st	1
Ex to 2nd	0
Mem to 2nd	0
Ex to 1st and Mem to 2nd	0

$$\text{New CPI} = 1 + 1 * (0.2) = 1.2$$

$$\textbf{Fraction of Stalled Cycles} = \frac{0.2}{1.2} * 100 = 17\%$$

Question 4.12.3

In the case of next-cycle forwarding, the fraction of stalled cycles is:

$$CPI_{NCF} = 1 + 1 * (0.1 + 0.1 + 0.2 + 0.05) = 1.45$$

Despite next-cycle forwarding, all the cases aside from EX to 1st still require at least a one cycle stall.

$$\text{Fraction of Stalled Cycles} = \frac{0.45}{1.45} * 100 = 31\%$$

In the case of two-cycle forwarding, the CPI is:

$$CPI_{NCF} = 1 + 1 * (0.05 + 0.2 + 0.1) = 1.35$$

$$\text{Fraction of Stalled Cycles} = \frac{0.55}{1.35} * 100 = 26\%$$

Hence, it is a better option to use two-cycle forwarding as there is a lower CPI, which leads to a smaller fraction of stalled cycles.

Question 4.12.4

In the equation below, the subscript 2 refers to the metrics of the pipelined CPU.

$$\text{Speedup} = \frac{T_1 * CPI_1}{T_2 * CPI_2} = \frac{590 * 1.85}{620 * 1.2} = 1.47 \tag{1}$$

We see that adding full forwarding makes the pipeline approximately 1.47 times faster than no pipelining.

Question 4.12.5

In the equation below, the subscript 2 refers to the metrics of the CPU with time-travel circuitry.

$$\text{Speedup} = \frac{T_1 * CPI_1}{T_2 * CPI_2} = \frac{620 * 1.2}{720 * 1} = 1.03 \tag{2}$$

We see that adding time-travel circuitry would make the new implementation 1.03 times faster than the full-forwarding implementation.

Question 4.14.1

empty space denotes n/a; ** denotes stalling

```
IF   ID   EX   MEM   WB
     IF   ID   **    EX    MEM   WB
                     IF    ID    EX    MEM   WB
                           IF    ID    **    EX    MEM   WB
                                 IF    **    ID    EX    MEM   WB
                                       IF    ID    EX    MEM   WB
```

Question 4.14.2

```
IF   ID   EX   MEM   WB
     IF   ID   **    EX    MEM   WB
          IF   **    ID    EX    MEM   WB
                     IF    ID    EX    MEM   WB
                           IF    ID    EX    MEM   WB
                                 IF    ID    EX    MEM   WB
                                       IF    ID    EX    MEM   WB
                                             IF    ID    EX    MEM   WB
```

Question 4.14.3

```
    lw r2, 0(r1)
label1:
    seq r8, r2, r0
    bnez, r8, label2
    lw r3, 0(r2)
    seq r8, r3, r0
    bnez r8, label1
    add r1, r3, r1
label2:
    sw r1, 0(r2)
```

Question 4.14.4

In order to support branching in the ID stage, we must account for three cases, each of which is a data hazard. Firstly, we must ensure that the previous instruction is not an R-type instruction that updates the result of any of the compared registers in the given branching instruction. If that is the case, then the branch instruction must be stalled for two cycles and then forwarded.

The second case refers to when a previous instruction is loading a value from memory into one of the registers being compared. In that case, it must be stalled 2 cycles so that the output of the MEM stage can be forwarded to the ID stage.

The third case is if two instructions above the current branch instruction is updating the register from data memory. Then, we must stall by one cycle to have access to the updated value through forwarding.

Question 4.14.5

We know that the original situation requires 14 clock cycles.

IF	ID	EX	MEM	WB									
	IF	**	**	ID	EX	MEM	WB						
				IF	ID	EX	MEM	WB					
					IF	**	**	ID	EX	MEM	WB		
								IF	ID	EX	MEM	WB	
									IF	ID	EX	MEM	WB

This requires 15 clock cycles. Hence, the speedup is $\frac{14}{15} = 0.933$, implying that moving branching to ID would be slower.

Question 4.14.6

If the branch instruction uses a register which was updated in the previous instruction, then the branch instruction must be stalled until the execution stage has been completed. We can then forward the output of the EX stage to the ID stage of the branch instruction.

Also, if the instruction prior to a branch instruction uses the data memory, then we can forward the value from the MEM stage to the ID stage of the next instruction.

The forwarding unit in Figure 4.62 requires the same inputs as those described above. Furthermore, the forwarding unit also provides select signals for two muxes. In the case of the new forwarading unit, it would also control select signals for two muxes. These muxes would be placed right before the comparator and would select what the comparator takes as inputs. Hence, the new forwarding unit would have the same complexity as the one shown in Figure 4.62.